41355031 计1302 马成宏

1. 设线性表La=(a0a1, ……,am-1), Lb= (b0b1, ……,bn-1)，利用线性表基本运算，求La – Lb =>La 、 La ∩ Lb =>Lc 运算的算法实现。
2. La – Lb =>La

算法思路：依次取Lb中的bi(i=0,1,…,n-1)，若bi在La中，则将其删除。

算法描述：

void Union(list \*La, list \*Lb)

{ int i,k; datatype x;

for(i=0; i<ListLength(Lb); i++)

{ x=GetElem(Lb,i); k=LocateElem(La,x);

if(k!=-1) Delete(La,k);

}

}

1. La ∩ Lb =>Lc

算法思路：依次取La中的ai(i=0,1,……,m-1)，若ai属于Lb,则将其存入表Lc中。

算法描述：

void Union(list \*La,list \*Lb,list \*Lc)

{ Int I,j,k=0;datatype x;

Clearlist(Lc);

For(i=0;i<Listlength(Lb);i++)

{ X=GetElem(La,i);k=LocatElem(Lb,x)

If(k!=-1)

{Insert (Lc,x,j); j++}

}

}

二．设学生记录表S ：（按学号Sno有序）

(1)设计表S 的顺序存储结构;

(2)写出将一学生记录x 插入到表中正确位置的算法:insert-s(S,x);

(3)写出从表中删除Sno=y 的记录: delete-s(S,y)。

(1)

#define maxsize 64

typedef struct //学生记录

{ int sno;

char name[20], sex, class[20];

}student;

typedef struct //表的类型

{ student s[maxsize]; //表的存储空间

int last; //当前表尾指针

}sqlist; \*sqlink; // 表声明符

(2)

int insert-s(sqlink ls, student x)

{ int i=0, j;

if (ls->last>=maxsize-1) return(0); //表溢出

while(i<=ls->last&&ls->s[i].sno<x.sno) //找x位置

{i++;}

if (ls->s[i].sno==x.sno) return(0); // x 已存在

for(j=ls->last; j>=i; j--) //记录顺移

{ ls->s[j+1]=ls->s[j];

ls->s[i]=x; //插入

ls->last++;}

return(1);

}

(3)

int delete-s(sqlink ls, int y)

{ int i=0, j;

if (ls->last==-1) return(0); //表溢出

while(i<=ls->last&&ls->s[i].sno!=y) //找y位置

{i++;}

if (i>ls->last) return(0); // y不存在

for(j=i; j<ls->last; j++) //删除sno=y 的记录

{ ls->s[j]=ls->s[j+1];

ls->last--;}

return(1);

}

三．试写出从表R中某p节点开始, 查找data=d的节点指针的算法：search(R,p,d).(算法前应包括对节点的说明)

typedef struct node //结点类型

{ datatype data; struct node \*next;

}linknode, \*link;

link search(link R,link p,datatype d)

{ link q=p;

if (R==R->next) return(NULL); //空表返回

do { if (q->data==d&&q!=R->next)

return(q); //找到

else q=q->next;

}while(q!=p)

}

return(NULL); //查找失败

}

四．写出判断A 表和B 表是否相等的算法：equal(A,B).

（两表相等的充分必要条件：表长相等，且两表中元素也对应相等。）

typedef struct node //结点类型

{ char data; struct node \*next;

}linknode, \*link;

int equal(link A,link B)

{ link p,q; p=A->next; q=B->next;

while(p&&q)

{ if (p->data!=q->data) return(0); //不等

else { p=p->next; q=q->next;}

if (p==NULL&&q==NULL) retuen(1); //相等

else return(0); //不等

}

}